Predicting Selective Availability for Instant Messaging

Mirko Fetter[1], Julian Seifert[2], and Tom Gross[1]

[1] HCI Group, University of Bamberg, Germany
`firstname.lastname@uni-bamberg.de`
[2] Paluno, University of Duisburg-Essen, Germany
`firstname.lastname@uni-due.de`

Abstract. Instant messaging (IM) systems allow users to spontaneously communicate over distance, yet they bear the risk for disruption of the recipient. In order to reduce disruption, novel approaches for detecting and presenting mutual availability are needed. In this paper we show how fine-grained IM availability predictions can be made for nomadic users solely based on sensors installed on a laptop computer. Our approach provides comparable accuracies to previous work, while it eliminates the need for augmenting the offices or the users with further sensors. We performed a user study to collect sensor data. Alongside with labels collected by means of Experience Sampling, the data allow for creating probabilistic models for predicting selective availability. This way, we demonstrate how the required effort involved in proactively managing one's availability selectively towards a variety of recipients can be reduced by automatic adaptation, and give insights in the lessons learned.

Keywords: Instant Messaging, Context Inference, Sensors, Privacy.

1 Introduction

Instant messaging (IM) systems offer a great possibility for people to spontaneously communicate over distance, yet bear the risk for disruption of the recipient. In order to reduce disruptions novel approaches for detecting and presenting mutual availability are needed. Previous research examined how predictions based on sensors installed in the environment and desktop computer can help a caller to better estimate, if a recipient in an office scenario is currently interruptible [1, 5, 6, 9]. Others have looked at predicting availability for phone calls based on body-worn sensors and sensors installed on mobile phones [8, 21]. In this paper we show how fine-grained IM availability predictions can be made for nomadic users solely based on sensors installed on a laptop computer. Our approach provides comparable accuracies to previous work, while it eliminates the need for augmenting the offices or the users with sensors, allowing it to be instantly applied in diverse environments and situations. This way, we demonstrate how the required effort involved in proactively managing one's availability selectively towards a variety of recipients can be reduced by automatic adaptation.

In this paper we first motivate the need for fine-grained predictions from a user-centred perspective since they allow for lightweight selective availability (i.e., easy management of different levels of availability to different groups of social contacts).

We elaborate on the technical and formal setup of the study and outline the underlying concepts for the collection of sensor data and user feedback. Further, we give detailed insight into the exploration, preparation, and analysis of the collected sensor data as well as into the machine learning performance. We identify lessons learned that can inform future research in this field and reflect our findings in comparison to related work. This paper concludes with an outlook to future work.

Accordingly, the contributions of this paper are: A demonstration of how machine learning can help to reduce the users' effort of managing well-differentiated selective availability in various environments, verified by the quality of the classification result; and a set of lessons learned that can inform future work in this field.

2   Automatic Adaptation of Selective Availability

Laptop computers give users the freedom to work at different places. Besides the office, people use them to work from home, at coffee shops or hotels, to take them to meeting rooms, presentations, or lectures, or to work on a train while commuting to the office or on a plane while flying to a business meeting. Such changing environments often come with different preferences to whom the user is currently available for communication. For communication via IM, presence and awareness information can help users to mutually present their availability for communication. However, with online status and status message most current IM systems only offer rudimentary support for their users to find an appropriate moment for initiating communication. Studies show that this deficit often leads to disruption of a recipient by an incoming message during a specific task or within an inappropriate situation resulting in communication breakdown [22]. Two fundamental challenges can be identified where current IM systems fall short:

- First, current IM systems only offer one single global online status. Therefore, users communicate their availability to all their contacts in the same way. A differentiation of their availability towards different categories of social contacts (e.g., family, colleagues, etc.) is only possible with workarounds in current systems. For example, Voida et al. [22] showed with a study that users maintain multiple IM accounts in order to achieve selective availability towards different categories of contacts. Other studies describe similar findings [2, 16].
- Second, the users need to adapt the online status manually. This leads to a self-interruption in their workflow and thus most times is simply forgotten. Hence, the resulting incorrect online status looses it significance for the users' contacts and accordingly is ignored because it is regarded as unreliable. This behaviour was also observed in the study of Voida et al. [22] were users textually manage their availability by enquiring. In their evaluation of a mobile 'Personal Presence' system Milewski and Smith [14] found that users changed their status on average only 1.4 times a day.

Based on these two premises, we conclude that managing selective availability by adapting multiple online statuses manually is not feasible for users. Therefore, we suggest automatic adaptation of the online status through predicting selective availability, which will help users to (a) maintain an up-to-date online status, (b) tailored towards different audiences, (c) while minimising their configuration effort.

In the following we demonstrate the feasibility of this approach by means of classification results with an average accuracy of 81.35% for the prediction of selective availability on the basis of real user data collected in a study.

3   Collecting the Data

We performed a user study in order to collect data and to research the predictability of fine-grained selective availability levels. The collected data, on the one hand, needs to describe the user's context. On the other hand, class labels that reflect the users inner states (i.e. availability preferences) are needed for training probabilistic models. Several related studies e.g., [8, 10] show that the Experience Sampling method in combination with sensor data logging is a suitable means for collecting such data.

In the following we illustrate the technical setup of this study and discuss in detail the selection of sensors and the Experience Sampling setup. Further, we report on the execution of the study and the collected data.

3.1   Technical Setup

Our study focuses on nomadic users where work and private life often intermix and therefore a need for selectively managing their availability exists. We used the primary tool of the study participants, their laptop computers, as platform for collecting data. On the one hand, recent laptops provide a variety of hardware sensors (e.g., camera, accelerometer), which can be used for detecting the user's context directly. Moreover, the computer usage already reflects a user's context considerably: the current used applications, the number of unread emails or the calendar entries all allow to make estimates about the users current context.

We implemented a system that runs as a daemon, drives a number of sensors, and saves lo-fi context data. A sensor is defined as a soft- or hardware component, or its combination that monitors and records the state of an artefact, an entity or the environment in the digital or physical realm. Sensors can be dynamically added as plug-ins, which are loaded at system runtime. The system provides a utility application for the investigator that allows the configuration of sensor parameters such as quantisation level or sampling rate (see Figure 1). The sampling rate for each individual sensor was balanced upfront in respect to expressiveness of the individual sensor and resource consumption and varied between 30 and 150 seconds.

By studying the related work, we came to a collection of sensors for capturing context data that worked in similar approaches [5, 6, 11]. In contrast to the related work, our approach aims at using only sensors that need no further instrumentation of the environment, in order to be applicable ad-hoc in different locations. Therefore, we discarded those sensors used in related work, which needed a stationary installation (e.g., door sensor, phone sensor) and tried to further exploit the capabilities of the mobile computers to gain more sensor information (e.g., build in accelerometer and photocell). All our sensors were implemented in Java and obtain their information either by making calls to native system libraries, by using command line calls or by running small scripts to access information from the hardware, the operating system, or specific applications.
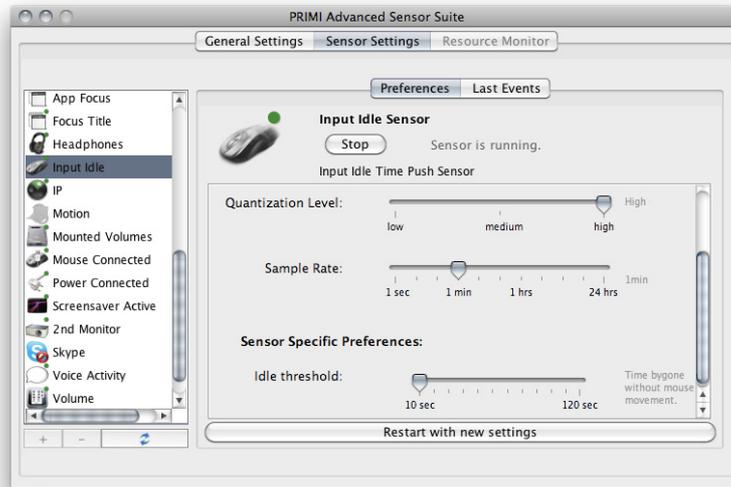
Fig. 1. Configuration dialog for adjusting the parameters of the sensors

In total we implemented 30 different sensors, each collecting one or more information chunks with each sensing, which where recorded in one of four different xml-data structures as a sensor event:

- *1×1 sensors*: return a single value per sensor event
- *1×n sensors*: return several, semantically different values of same or different types per sensor event
- *n×1 sensors*: return a list of values of the same type
- *n×m sensors*: return a matrix of values

In the following list all implemented sensors and their capabilities are described:

- Active Access Point (1×1): ID of the access point the computer is connected to
- Active Chats (1×1): number of active chat windows for all used IM applications
- Active Network Interfaces (n×1): list of active network interface IDs
- Ambient Light (1×1): intensity level of ambient light
- Applications (n×1): list of running applications
- Application Focus (1×1): name of the application in focus
- Battery (1×n): current battery capacity; charging state; being fully charged
- Bluetooth Devices (n×1): list of nearby Bluetooth devices
- Calendar (1×n): whether there are upcoming events; whether an event is ongoing
- Connected FireWire Devices (n×1): list of IDs of connected FireWire devices
- Connected USB Devices (n×1): list of IDs of connected USB devices
- CPU (1×n): user CPU load (%); system CPU load (%); idleness of CPU (%)
- Email (1×n): number of unread; number of received; and number of send email
- Ethernet Connected (1×1): is computer connected to a network by a Ethernet cable

- Face Detection (1×1): is a face detected on the image of the built-in web cam
- Focus Title (1×1): title of the front most window
- Headphones Connected (1×1): is external audio hardware connected
- Input Idle (1×1): are keyboard and mouse idle for a specific period
- IP Address (1×n): External IP address (given by the ISP); Local IP address
- Mounted Volumes (n×1): list of mounted volumes
- Motion (1×n): acceleration of X-axis; acceleration of Y-axis; acceleration of Z-axis
- Mouse Connected (1×1): whether a mouse is connected
- Power Connected (1×1): whether the power cable is plugged in
- Screensaver Active (1×1): whether the screensaver is active
- Second Monitor (1×1): number of connected monitors
- Skype (1×n): number of online contacts; mood message; status; call is ongoing
- Time (1×n): hour of the day; day of the week; weekend; part of day
- Voice Activity (1×1): is human voice detected via the built-in microphone
- Volume Settings (1×n): output volume; alert volume; audio is muted
- Wi-Fi (n×m): list of nearby Wi-Fi networks each with: SSID, BSSID, and RSSI

3.2   The Experience Sampling Dialog

In addition to this list of sensors an Experience Sampling sensor was installed. Its purpose was to prompt participants for an assessment of their current availability. These self-estimates serve as label for the succeeding training and evaluation of probabilistic models by means of machine learning. This sensor presented a dialog (see Figure 2) in which participants were prompted to give a statement about their current general availability and their selective availability as well as their current location. General availability was defined as the availability towards all of their contacts—as they would set it in their current IM system. Selective availability was measured for three *availability categories*: *Private*, *Work,* and *Public/Others*. We defined *availability category* as a group of contacts towards which the user is available in the same way.

In order to allow for comparison between the participants, we pre-specified three availability categories instead of letting the participants choose their own arbitrary number of availability categories. These are based on a pre-study in which we looked at research focussing on privacy and sharing preferences in the context of IM as well as in general awareness support (e.g., [7, 15, 17, 18]). Based on these studies we identified seven common categories for grouping contacts, towards which information is disclosed in the same way (e.g., family, team, subordinates, etc.). In order to reduce the effort for the Experience Sampling, we conducted a paper-based questionnaire survey to trim down the number of categories. We asked 30 participants to group the identified categories into optionally three or four clusters according to how they would be available for theses groups of persons. The configurations that occurred most incorporated three clusters, which make up our three availability categories and were labelled *Private*, *Work*, and *Public/Others*.

Within these categories the participants could assess their availability in terms of six *availability levels: Offline*, *Do not disturb*, *Not available*, *Away*, *Online*, and *Text Me!*. We chose these six availability levels, imitating the online statuses provided by

Fig. 2. Experience Sampling interface for collecting user selective availability preferences

Skype, as all our participants are frequent Skype users and thus used to these levels. We preferred this over using a five-point Likert-scale or a simple binary assessment of "Available" versus "Unavailable".

During the study, this Experience Sampling dialog was presented every 25 to 35 minutes as the front-most window. When the participants did not start to interact with the dialog for more than 30 seconds, the dialog disappeared and reappeared 5 minutes later. In the moment participants started interacting with the dialog, this countdown was stopped, until the participants completely filled out and submitted the dialog. Afterwards the dialog disappeared and started over to appear in the normal interval. This way, we assured that we don't miss a number of samplings when the participants are often away from screen for a short time.

3.3 Study Setup and Execution

While ESM allows collecting rich and in-depth data through repetition the disruption and the resulting effort for the individual subjects during a four-week study is very high. In order to achieve comparable results we used the same number of subjects as in related work [5, 11] in order to collect a similar amount of data. We therefore recruited four volunteers as participants (P1-P4) for a four-week long study. The participants were all males and aged between 25 and 33 (M=28.0). Three participants have a background in computer science one participant in social science: two participants were research assistants and the other two participants were student assistants. All come with several years of experience with instant messaging (M=9.5; SD=4.5) and use in average 4.5 (SD=2.0) different instant messaging accounts; all participants use Skype. Except one participant, who indicated to use instant messaging only during workdays, the others use instant messaging on a daily basis. All participants indicated that they use their laptop at different locations during a normal workday. The study was framed by a short questionnaire upfront and a semi-structured interview afterwards.

Data on the participants were collected over the period of four weeks by means of the sensor daemon installed on their laptop computers. The participants were briefed to run the sensor daemon continually and to turn it off only in exceptional situations (e.g., when holding a presentation or the computer's battery runs low). Therefore, each participant collected a different amount of sensor samples and Experience Sampling self-reports. Figure 3 shows the difference between cast and actually answered self-reports.

**Overeiew of cast and answered experience samples**
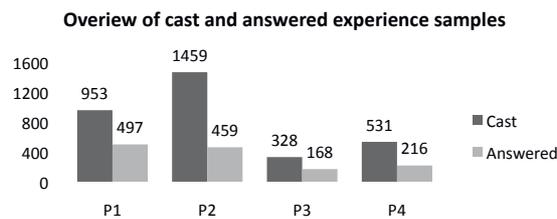
Fig. 3. Amounts of cast and actually answered self-reports for each participant

On average, the Experience Sampling dialog was presented 817.75 times (SD=500.52) and 270,726 (SD=133,612) sensor events were collected by 30 sensors. These numbers (c.f. Figure 3) show that the majority of the dialogs remained unanswered. This is an intentionally introduced bias of the study design. As described in 3.2, as long as a dialog remains unanswered the sampling interval is decreased from 30 to 5 minutes in order to miss less data when the computer is left unattended only for a short period. On the other side this leads to a high number of unanswered dialogs for longer breaks (e.g. a one-hour lunch break would result in 12 instead of 2 unanswered dialogs), which can be neglected. An approach to treat unanswered dialogs as an indicator for unavailability was discarded, as it is unclear whether the subject is unreceptive or really not available [8].

Overall participants gave their estimates concerning their availability in average 355 (SD=167.00) times. The differences between the subjects illustrated in Figure 3 are likely due to the varying periods the sensor daemon was running as well as different computer usage patterns. For instance, P4 had the software installed on a computer usually being used only during common working hours. P3, who answered the smallest amount of samples, had the sensor daemon installed on a personal laptop computer but turned it off frequently because off the performance loss it caused. P1 and P2 ran the daemon in average 336 hours during the four-week study period. The high number of unanswered self-reports of P2 could be caused by the individual usage pattern letting the computer run while being physically away from it. The samples were collected at 4.8 (SD=2.2) locations including the respective users' office, meeting rooms, lecture rooms, other people's offices, the university' cafeteria or at home. A more qualitative interpretation of the Experience Sampling data in respect to the participants' needs for selective availability is discussed in [3]. In the following we will solely focus on the collected data from the perspective of predictability of selective availability through machine learning.

4   Predicting Selective Availability

In this section we illustrate the pre-processing of the data and to what extend it is possible to predict selective availability levels based on collected sensor data. We describe the process of transforming the sensor data into features, the generation and selection of features, and the selection of the classification algorithms. This proceeds with the discussion of the evaluation of the machine learning performance.

4.1   Feature Generation, Selection, and Classification

The data provided by sensors do not allow reasoning about the users' availability right away. It is necessary to transform the collected sensor data logs from each user into meaningful separated features. For instance, the *Applications*-sensor ($1{\times}n$) provides a list of all running applications with each sensor reading. Each application name contained in the sensor value refers to one feature, which needs to be extracted. Hence, when creating a probabilistic model all the lists of application names need to be processed to collect the complete list of available features.

Besides this simple transformation of sensor data into features, we generated additional features by integrating the feature values over time. This allows to take advantage of a sensor's value history which reveals trends and tendencies. In order to keep the number of generated features low, we decided to generate time integrations only for five, eight, and 14 minutes. In particular, for numeric features we calculated the mean and standard deviation for the respective history time. For Boolean values we calculated to what extend the particular feature was true or false during the respective period. This approach of feature generation increases the dimension of the feature space enormously. The next step therefore was to discard features with a minimal information gain upfront. The resulting number of features after this reduction still was 1437.3 in average.

As most machine-learning algorithms perform better in a low dimensional feature space, it was necessary to further reduce the number of features for the classification. We compared different techniques for selecting a small subset of features. In particular, we focused on Correlation-based Feature Selection (CFS), Information Gain, Gain Ratio, and Chi-square Feature Subset Selection. Since different classifiers perform significantly different when trained with differing feature subsets, it is necessary to explore the selection of the classifier in combination with the feature subset-selection. Therefore, we considered four different classifiers: Naïve Bayes, C4.5 Decision Tree, Random Trees, and Support Vector Machine (SVM). We used the machine-learning workbench WEKA [23] and its experimenter tool for determining which subset of features in combination with which classifier leads to the best classification results. The results show that WEKA's SMO Classifier—a SVM implementation utilising the Sequential Minimal Optimization (SMO) [19] algorithm for training—in combination with a feature subset selected by means of Correlation-based Feature Selection (CFS) outperforms all other combinations. Subsequently, the best parameters for the SMO algorithm were determined using WEKA's GridSearch for further optimising the classification performance.

4.2   Evaluation of Machine Learning Performance

The evaluation of the predictability of selective availability is based on the data collected in the user study. The evaluation consists of building four probabilistic models for each user using the SMO algorithm based on the (in average 23.4 (SD=10.8)) features selected by the CFS in combination with the (in average 355 (SD=167.00)) samples based on the labels given by the individual participants. For each participant, four models were built—one for *General availability* and three for the availability categories. Each model is trained to predict the six availability levels, which makes the classification task a multi-class problem. The 10-fold cross-validated classification results for the 4x4 models are summarised in the following Table 1.

Table 1. Results of 10-fold cross-validations for each of the four participants' models, and the average over all models of all participants. Each table presents the accuracy in percent for the ZeroR classifier (to show the base probability), the SMO classifier, and the improvement of the SMO classifier against ZeroR classifier (Diff.) for the different models.

Model	ZeroR	SMO	Diff.
P1 General	58.15	75.65	17.51
P1 Private	55.53	75.86	20.32
P1 Public	59.76	78.27	18.51
P1 Work	51.31	75.86	24.55
P1 Average	**56.19**	**76.41**	**20.22**

Model	ZeroR	SMO	Diff.
P3 General	74.40	91.67	17.26
P3 Private	86.90	92.26	5.36
P3 Public	69.64	86.90	17.26
P3 Work	78.57	92.26	13.69
P3 Average	**77.38**	**90.77**	**13.39**

Model	ZeroR	SMO	Diff.
P2 General	68.62	84.10	15.48
P2 Private	70.29	83.68	13.39
P2 Public	66.53	82.01	15.48
P2 Work	71.13	84.94	13.81
P2 Average	**69.14**	**83.68**	**14.54**

Model	ZeroR	SMO	Diff.
P4 General	62.50	74.54	12.04
P4 Private	70.37	76.85	6.48
P4 Public	74.54	76.85	2.31
P4 Work	64.35	69.91	5.56
P4 Average	**67.94**	**74.54**	**6.60**

Model	ZeroR	SMO	Diff.
All Average	**67.66**	**81.35**	**13.69**

In the most left column of each table the name of the respective data set is listed. In the second column, the base probabilities generated by the ZeroR algorithm are listed. In the third column the accuracy based on a stratified 10-fold cross-validation of the models that were built by the support vector machine SMO are listed. The last column contains the difference between base probability, as estimated by the ZeroR classifier, and the percentage of correctly classified instances by the SMO. The average over the four models of each participant reflects the overall experience the users will have, as all four models will work in unison to adapt the users selective availability.

The results show that over all participants the base line is quite high with 67.66%. This means, in average, one online status is selected more often then all other online status together by a user for a specific availability category. Accordingly, a classifier

that always guesses this dominant class would make a correct prediction in 67.66% of the cases. Therefore, the results from the evaluation of the probabilistic model need to be regarded relative to their respective base line. The results shows that in average the probabilistic models perform considerably better (13.69%) than the base line. Overall, with an average accuracy of 81.35% our results can compete with the results of related work; as discussed in detail later.

5   Lessons Learned

In the following section we elaborate on lessons learned during the study and the explorative machine learning procedure, which can inform future research on similar approaches. We elaborate on the performance of the individual sensors for machine learning. We give guidelines on the engineering of robust sensors for this kind of studies. We motivate our argumentation for building personalised models for each individual user. We describe our findings on the usage of the Experience Sampling method. And we show patterns we found in the configuration of selective availability.

5.1   Promising Sensors and Features

The results of the feature selection mechanisms allow drawing conclusions on the relevance of single sensors for the prediction result. Such knowledge can inform future research at selecting relevant sensors. Therefore, we analysed the features of all 16 models (four participants, each with a model for General, Private, Work, and Public/Others) that were selected by the correlation-based feature selection algorithm as most relevant. CFS uses a heuristic evaluation method and has proven to be equally good at finding relevant features in machine learning tasks in comparison to wrapper-based approaches. Hence, the selected subset can give a valid estimate on the contribution of individual sensors.

The CFS algorithm reduced the number of features from 1437.3 (SD=303.6) per model to the 23.4 (SD=10.8) most relevant. First, we consolidated the reduced feature sets of all 16 models and analysed what feature generation mechanism delivered the largest number of features to this set. 71.93% of the features were generated by the time-based feature generation. The features that take into account the last 14 minutes made up the majority with 32.89%, followed by 5 minutes (21.39%) and 8 minutes (17.65%). As expected, taking into account the sensor values for a period of time instead of only a point in time can deliver richer features for the classification [4].

Further, we analysed the sensors providing one or more relevant features to one or more of the four models for each participant according to the result of the CFS. Table 2 shows that while eleven sensors generated features relevant for at least one model of each of the participants, ten sensors did not. For the category of unused sensors, we found that the users did not influence some measured parameters during the four-week study period. For example, none of the users connected FireWire devices or mounted a volume. Accordingly, these sensors can be valuable for users that use such devices or services more often. Other unused sensors, like the Mouse Connected Sensor might be substituted by the information of a second sensor, like the Connected USB Devices sensor. Further, our expectation that different combinations of sensors

Table 2. Sensors that delivered features to the final feature subset: "+" sensor delivered one or more features , "-" sensor delivered no feature for this participant's models.

Sensor	P1	P2	P3	P4	No.		Sensor	P1	P2	P3	P4	No.
Active Access Point	+	+	+	+	4		Focus Title	+	+	+	+	4
Active Chats	-	-	-	-	0		Headphones Connected	-	-	-	-	0
Active Network Interfaces	-	-	-	+	1		Input Idle	-	+	+	+	3
Ambient Light	-	+	+	+	3		IP Address	+	+	+	+	4
Application Focus	+	+	+	+	4		Motion	+	+	-	+	3
Applications	+	+	+	+	4		Mounted Volumes	-	-	-	-	0
Battery	-	-	-	-	0		Mouse Connected	-	-	-	-	0
Bluetooth Devices	+	+	+	+	4		Power Connected	+	-	-	-	1
Calendar	+	+	+	-	3		Screensaver Active	-	-	-	-	0
Connected FireWire Dev.	-	-	-	-	0		Second Monitor	-	-	-	-	0
Connected USB Dev.	+	+	+	+	4		Skype	-	+	+	+	3
CPU	-	-	+	+	2		Time	+	+	+	+	4
Email	-	-	-	-	0		Voice Activity	+	+	+	+	4
Ethernet Connected	-	-	-	-	0		Volume Settings	+	+	+	+	4
Face Detection	-	+	+	+	3		Wi-Fi	+	+	+	+	4

monitoring cables attached to the computer (Ethernet, Headphones, Power, etc.) would give a good estimate about the current location, were not met, as theses sensors practically played no role for the models.

However, most of these unused sensors are not computationally or energetically expensive, as they basically listen to system events, in comparison to more expensive operations like constantly polling for available Wi-Fi networks or performing face-detection on the camera image. As there is a chance that these sensors might be useful for other users, it might be reasonable to not discard them completely. Further, some of these sensors might be valuable for indicating transitions between different contexts. As our Experience Sampling study used a random interval of circa 30 minutes to assess the users' availability, the collected samples cannot reflect the exact moment when users' inner states change e.g., from Available to Do Not Disturb. Some of the sensors that were not used for building the predictive models could be valuable for predicting a good moment for adapting the online status. For example, disconnecting the mouse and Ethernet from a mobile computer could indicate that the user is leaving the current place, and suggest a change in availability.

5.2 Carefully Engineered Sensors

The design and implementation of the sensors and the surrounding architecture are crucial for the success of the study. In the following we list several technical requirements, which we found in small informal pre-tests and during the study.

The sensors for data collection need to be robust. During the study the aim was to continuously collect data while the system was in normal use. Accordingly, the implementation should be robust to system restarts, idle-times, crashes or

reconfigurations while not limiting the users in their tasks. Crashing sensors should automatically recover without the need for user intervention. We implemented mechanisms that monitor and automatically restart sensors that stopped delivering values. At the same time, sensors should not interfere with user interactions. Sensors that exclusively allocate system resources (in our case camera and microphone) need to be designed in away that these resources are still available for the users' primary tasks (in our case for a video chat). In our setup the sensors obtained data and were released for each sampling, with small pauses between the samplings. If a resource was occupied, the sensors tried again until the resource was obtainable (e.g., after the video chat was finished). In the same way, sensors should not open applications like calendar or email to retrieve sensor values, unless the users open them.

In addition to this, collecting sensor data continuously from a mobile computer leads to a reduced battery lifetime and system performance. Continuously accessing hardware like the camera or the embedded acceleration sensors in combination with repeated polling for nearby Wi-Fi or Bluetooth networks has an unavoidable but substantial effect on the battery and CPU. In order to reduce this drawback, we balanced the sampling intervals for the different sensors between maximising the amount of valuable data that is collected and minimising usage constraints.

Further, privacy-protecting mechanisms have a major effect on the users' willingness to collect data. We implemented a hash-based privacy based on the Subtle system [4]. Each nominal value that was collected (e.g., the title of the current focussed application, the SSID of nearby Wi-Fi access points) was translated into a unique hashed representation. In this way the data is still meaningful for machine learning algorithms, but reduces its descriptiveness for humans, who prepare the data for machine learning. However, this comes with a decreased interpretability for reflecting on more qualitative aspects of the data in an analysis of the study results. For example, we are unable to make statements about availability preferences while using specific applications like Word or Firefox—which could provide interesting insights for CSCW research—due to the encryption.

Finally, the data collection process needs to be as unobtrusive to the users as possible, yet has to allow the users to stay in control. Whether for reasons of system performance or for privacy reasons, the user has to be able to pause the data collection at any time. In our system, users were able to stop and start the collection with the click on a button. However, in order to reduce the users' need to do so, and to obtain more data from the study, the above guidelines can help.

5.3 General Models vs. Individual User Models

One drawback of building an agent-based system is that the system needs to learn from scratch and so is of limited utility to the user at the beginning or in new situations. In order to overcome this issue Collaborative Interface Agents [13] were introduced that share their knowledge among user models. In the field of predicting human interruptibility Fogarty et al. [6] for example built one general model for all subjects and general models for each group of subjects (manager, researcher, intern) in order to overcome the cold-start problem. However, their individual group models performed better than the global general model. From our results we can confirm this findings and even advocate for completely personalised individual models.
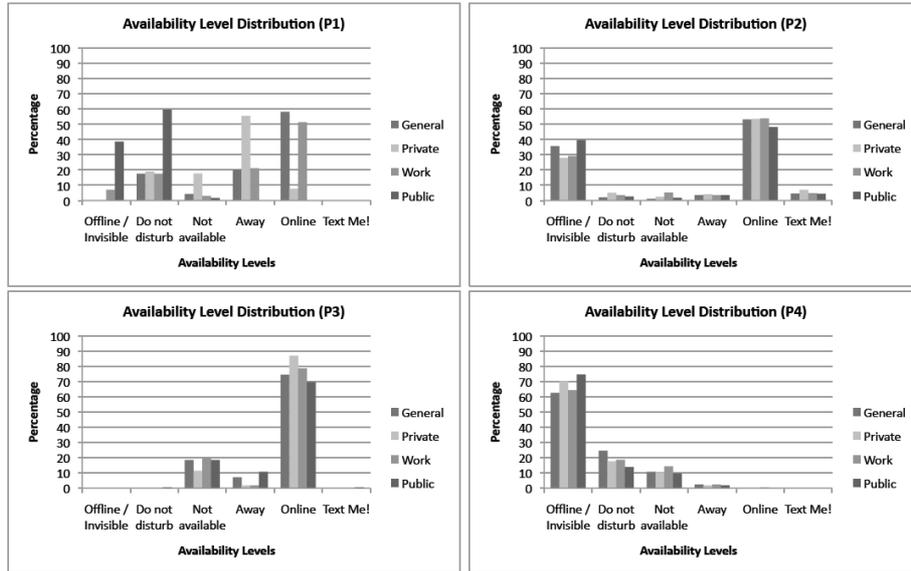
Fig. 4. Selective Availability distributions for the four subjects in percent

The first reason is that we found that even among the small number of our subjects, the individual availability expressed through the chosen availability levels strongly varied as can be seen in Figure 4. The Figure shows that while P3 had a clear tendency to being highly available, P4 has a clear tendency to be unavailable, while P1 and P2 have a broader spectrum of availability. This argues for the users' individual understanding of and a personal perspective on availability that a general model can hardly reflect.

Further, such general models need to be built solely on features that all users have in common. This bears three challenges: First, a feature with a high informative value for an individual model, like a wireless network at home, would most likely be discarded from the feature vector of a general model, as it would contain missing values for the other subjects. Second, some of the features may represent contradictory information for different users. Referring to wireless networks as an example, the wireless networks that are accessible from a certain place, may indicate the office for one user and a meeting room for another user, and accordingly different availability preferences. Third, some features' values might correlate very well for different users, but cannot be automatically merged into one feature. For example, the use of the browser or a word processor might lead to similar availability preferences for different users. However, when different users use different applications (e.g., one user uses Firefox, the other user Opera) this sensed information cannot be combined easily. Of course, all three challenges could be overcome by introducing a semantic layer between the sensed data and the generated features. But in many cases (e.g., mapping the Wi-Fi Access Points at home to the semantic representation home) requires further effort by the users for generating individual semantic abstractions.

Accordingly we found for our data: The performance of general models that were built with different classifiers on the intersecting subset of features from all the collected data performed insignificantly better than the base probability of the ZeroR classifier, but most times worse. Satisfactory classification results could not be achieved by using a general model.

5.4   Designing the Experience Sampling

The Experience Sampling dialog is the predominant interface to participants and needs special care. During the study the users are interrupted by the dialog several times a day during their daily tasks. In order to keep the study participants motivated to deliver valid self-reports, it is necessary to minimise the effort for the users on the one side and to prevent habitual, repeated responding [20] on the other side.

The design of our interface was optimised in iterative steps. It minimises the number of clicks for users, and prevents a bias towards repeated responses. We abandoned a design, where the choices from the previous sampling were pre-selected, as this resulted in a bias towards just confirming the previous choices. A design where we randomised the ordering of the items each time dramatically increased the users' cognitive effort and accordingly increased their frustration with the study—and so was also discarded.

In order to minimise the variance of free-text answers for the "Current location"-field, we implemented an auto-complete mechanism. From the subsequent interviews we learned, that this sometimes led to an automatism to accidentally enter and select the location that the users stayed at the most (e.g., their office). This came with a desire of the users for a mechanism to recall and alter or discard the last assessment.

The dialog always appeared as the front-most window in the centre of the screen and could not be discarded by the user. This rather invasive design helped to maximise the number of samples, since the fastest way to get rid of the dialog was to quickly respond to the questions. In the interviews we learned that this was perceived as acceptable in situations where the participants directly interacted with the computer, but was regarded as problematic where the computer was used more indirectly (i.e. to watch a video or to give a presentation). One participant suggested a configuration, which would allow shifting the Experience Sampling to a mobile phone for a certain period of time.

Further, the timing of the Experience Sampling plays a crucial role. We decided to show the dialog at a random interval, with one exception: Every time the users logged in, a sampling was triggered. During the study and afterwards, from the data, we learned that there are a lot of simple cues (that our sensors are able to measure) which can be easily formulated as plain rules to allow sampling at a moment, that typically marks the transition between different availability levels: when the computer is moved (Motion-Sensor), cables are attached or detached (e.g., Power Connected-Sensor, Ethernet Connected-Sensor) or someone starts to speak (Voice Activity-Sensor). These are a few examples for such moments that show how a context-aware Experience Sampling approach [12] could help to improve the quality of the data.

In combination with the sensor logging, the time between the presentation of the dialog and the response have to be taken into account. As the presentation of the

dialog (Application Focus-Sensor) and the users interaction with the dialog (Input Idle-Sensor) have an influence on what the sensors measure, they introduce a systematic error. Therefore it is necessary to discard the sensor data from the moment before the dialog is shown until the response was entered. If the time between presentation and response is to long, the sensor data and the Experience Sampling data will loose its correlation. In our design the user had 30 seconds to answer to the Experience Sampling before it disappeared, and was presented again five minutes later. In these 30 seconds no sensor data was recorded.

Finally, users showed great interest in the data collected when we discussed it in the subsequent interviews. To make their data available after the study to each of them individually in a meaningful way (e.g., as charts or diagrams) could make a great incentive to keep them motivated throughout the study.

5.5 Patterns for Selective Availability Configurations

The analysis of the collected data revealed that users tended to give similar answers in reoccurring contexts. For example during work-time a user might be available for work related contacts, less for private, and offline for others. This raises the question, how many different patterns or configurations users utilise to state their selective availability in the majority of the times.

We used the expectation-maximisation-algorithm for finding the patterns that fit best to the users' answers. The following table presents the results.

Table 3. Patterns (a-d) of availability, by means of clustering the participants ESM data (1=Offline, 2=Do not disturb, 3=Not available, 4=Away, 5=Online, and 6=Text Me!)

	P1				P2				P3			P4			
	a	b	c	d	a	b	c	d	a	b	c	a	b	c	d
Private	4	5	3	2	5	6	4	1	5	4	3	1	2	3	4
Public	2	2	1	1	5	6	1	1	5	4	3	1	2	3	4
Work	5	1	4	2	5	6	4	3	5	5	3	1	3	3	4

For three participants, four patterns were identified (a, b, c, d). For P3 only three patterns (a, b, c) were identified. These patterns respectively make up 84.32% (SD=3.40%) of the used configurations. On average the participants only used 23.25 (SD=14.97) combinations of the 216 possible permutations of availability levels and availability categories.

Since the numbers of the patterns for each participant are smaller than the number of all possible permutations, the complexity of the resulting classification problem is reduced. After transforming the data, again, we created probabilistic models, which were evaluated in terms of their ability to predict the correct pattern when given an example. The results showed, that the individually trained models for each availability category significantly outperform the models for predicting patterns.

Even though the results in this case are discouraging, the insight in such patterns can inform the design of future studies and of instant messaging systems.

6   Related Work

We briefly discuss related work and outline similarities and differences to our work. To our knowledge, so far no research has focussed on the predictability of selective availability based on sensors on laptop computers. We concentrate on work that focuses on predicting the availability for phone calls based on mobile sensors and two prominent examples that deal with the predictability of general interruptibility based on sensor data in office environments, which inspired our approach.

Ter Hofte [21] used the software SocioXensor for PDAs and mobile phones to collect data from 10 participants over 7 days via ESM. The resulting predictions are not explicitly focused on selective availability. These predictions—based on answers to the ESM (in conversation, location, etc.) and not on real sensor information—only achieve an accuracy of 63,9% for a two-class problem (green light/no green light). Ho and Intille [8] use activity recognition based on wearable accelerometers and a decision tree classifier on a PDA to detect postural and ambulatory transitions. They found that transitions are a good indicator for receptiveness towards interruptions. Their approach burdens the user to wear accelerometer sensors at ankle and thigh.

Fogarty et al. [6] use a combination of computer (mouse and keyboard activity, used applications, etc.) and room sensors (door, telephone, motion, etc.) to predict the interruptibility of 10 participants. They collected 975 oral interruptibility self-reports with interruptibility on a five-point Likert-scale. They used a naïve Bayes classifier to build three models for the three different types of participants and one general model for all participants. The achieved accuracies lie between 80.1% to 87.7% for the three models and 79.5% percent for the general model. However, they reduced the complexity of the prediction to a two–class problem (1-4 and 5) to obtain these results. In comparison, our average accuracy for predicting a six-class problem based on individual models lies at 81.35%. Their room sensors limit its applicability. Our participants collected data in several different locations in work and private contexts, which also introduces a greater complexity to the prediction task.

Also the BusyBody system [11] relied on sensor data collected on computers in an office environment. Four participants collected between 789 and 2365 assessments of their interruptibility ("Busy" and "Not Busy"). The average achieved accuracy with a Bayesian Network is 78.25%. Their prediction is also only for a binary classification problem. There is no information given, in which contexts the data was collected.

Concluding, none of the related work focussed on predicting selective availability for sensors on laptop computer. However, all the works greatly informed our research. Our aim was to combine the advantages of stationary and mobile approaches and to deepen it, in order to show that even more complex problems can be addressed with this approach while minimising the setup effort for deploying additional sensors.

7   Conclusions and Future Work

We have shown that the prediction of selective availability with machine learning is a feasible approach. Based on our results we conclude that systems that rely on sensor data collected on a laptop computer can be trained by users in order to allow for the automatic adaptation of selective availability in Instant Messaging. Such systems

support users in better managing their availability towards different social contacts while at the same time minimising their effort in comparison to manual adaptation.

We showed how this is possible by information that is directly accessible on the users' laptop computer. We showed that we can predict, with an accuracy of 81.35%, in which way users are available for different social contacts in different situations. This was shown for work and private contexts. And it allows users to express availability with fine-grained online statuses.

In order to evaluate how this approach gives a greater flexibility to users while minimising their effort, user tests based on a system that provides such predictions in real-time are needed. To support individual availability categories for each user such systems have to support life-long learning to train an arbitrary number of individual models for each user (i.e. continuously and actively learn from the users' inputs as well as predict to adapt the system in the background). To achieve this, such systems should automatically extract, generate, and select useful features from sensor-data streams. At the same time they should request input from the users in a much more implicit way.

References

1. Begole, J.B., Matsakis, N.E., Tang, J.C.: Lilsys: Sensing Unavailability. In: Proceedings of ACM CSCW 2004, pp. 511–514 (2004)
2. Davis, S., Gutwin, C.: Using Relationship to Control Disclosure in Awareness Servers. In: Proceedings of GI 2005, pp. 145–152 (2005)
3. Fetter, M., Seifert, J., Gross, T.: Lightweight Selective Availability in Instant Messaging. In: Extended Abstracts of ACM CHI 2010, pp. 3817–3822 (2010)
4. Fogarty, J., Hudson, S.E.: Toolkit Support for Developing and Deploying Sensor-Based Statistical Models of Human Situations. In: Proceedings of ACM CHI 2007, pp. 135–144 (2007)
5. Fogarty, J., Hudson, S.E., Akteson, C.G., Avrahami, D., Forlizzi, J., Kiesler, S., Lee, J.C., Yang, J.: Predicting Human Interruptibility with Sensors. ACM Transactions on Computer-Human Interaction (TOCHI) 12(1), 119–146 (2005)
6. Fogarty, J., Hudson, S.E., Lai, J.: Examining the Robustness of Sensor-Based Statistical Models of Human Interruptibility. In: Proceedings of ACM CHI 2004, pp. 207–214 (2004)
7. Gross, T., Fetter, M.: Disclosure Templates for Selective Information Disclosure. In: Proceedings of IADIS CT 2010, pp. 101–108 (2010)
8. Ho, J., Intille, S.S.: Using Context-Aware Computing to Reduce the Perceived Burden of Interruptions from Mobile Devices. In: Proceedings of ACM CHI, pp. 909–918 (2005)
9. Horvitz, E., Apacible, J.: Learning and Reasoning about Interruption. In: Proceedings of the 5th International Conference on Multimodal Interfaces - ICMI 2003, pp. 20–27 (2003)
10. Horvitz, E., Kapoor, A.: Experience Sampling for Building Predictive User Models: A Comparative Study. In: Proceedings of ACM CHI 2008, pp. 657–666 (2008)

11. Horvitz, E., Koch, P., Apacible, J.: BusyBody: Creating and Fielding Personalized Models of the Cost of Interruption. In: Proceedings of ACM CSCW 2004, pp. 507–510 (2004)
12. Intille, S.S., Rondoni, J., Kukla, C., Ancona, I., Bao, L.: A Context-Aware Experience Sampling Tool. In: Extended Abstracts of ACM CHI 2003, pp. 972–973 (2003)
13. Lashkari, Y., Metral, M., Maes, P.: Collaborative Interface Agents. In: Proceedings of the Twelfth National Conference on Artificial Intelligence - AAAI 1994, pp. 444–449 (1994)
14. Mileswki, A.E., Smith, T.M.: Providing Presence Cues to Telephone Users. In: Proceedings of ACM CSCW 2000, pp. 89–96 (2000)
15. Olson, J., Grudin, J., Horvitz, E.: A Study of Preferences for Sharing and Privacy. In: Proceedings of ACM CHI 2005, pp. 1985–1988 (2005)
16. Patil, S., Kobsa, A.: Instant Messaging and Privacy. In: HCI 2004: 18th British HCI Group Annual Conference. pp. 85–88 (2004)
17. Patil, S., Kobsa, A.: Uncovering Privacy Attitudes and Practices in Instant Messaging. In: Proceedings ACM SIGGROUP 2005, pp. 109–112 (2005)
18. Patil, S., Lai, J.: Who Gets to Know What When: Configuring Privacy Permissions in an Awareness Application. In: Proceedings of ACM CHI 2005, pp. 101–110 (2005)
19. Platt, J.C.: Fast Training of Support Vector Machines Using Sequential Minimal Optimization. In: Schölkopf, B., Burges, C.J.C., Smola, A.J. (eds.) Advances in Kernel Methods - Support Vector Learning, pp. 185–208. MIT Press, Cambridge (1998)
20. Scollon, C.N., Kim-Prieto, C., Diener, E.: Experience Sampling: Promises and Pitfalls, Strengths and Weaknesses. Journal of Happiness Studies 4(1), 5–34 (2003)
21. Ter Hofte, G.H.: Xensible Interruptions from Your Mobile Phone. In: Proceedings of MobileHCI 2007, pp. 178–181 (2007)
22. Voida, A., Newstetter, W.C., Mynatt, E.D.: When Conventions Collide: The Tensions of Instant Messaging Attributed. In: Proceedings of ACM CHI 2002, pp. 187 – 194 (2002)
23. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, San Francisco (2006)